Optimization and Analysis of Force Field Parameters by Combination of Genetic Algorithms and Neural Networks

J. HUNGER, G. HUTTNER

Anorganisch-Chemisches Institut der Universität Heidelberg, Im Neuenheimer Feld 270, D-69120 Heidelberg, Germany

Received 28 August 1998; accepted 2 November 1998

ABSTRACT: In search of a force field description for *tripod* metal templates, *tripod*M [*tripod* = $RC(CH_2X)(CH_2Y)(CH_2Z)$; X, Y, Z = PR′R″] force field parameters, p, were optimized by the use of genetic algorithms (GA) with the structures of ten compounds, *tripod*$Mo(CO)_3$, serving as the database. It was found that the evaluation of the fitness criterion, based on the root-mean-square deviation (rms) between observed and calculated structures by force field methods, is actually the time-consuming step under this optimization protocol. It is shown now how this time-consuming step may in part be substituted by using a trained neural network (NN) as the evaluating function. The network is trained on the basis of parameter vectors that have been evaluated previously with respect to their corresponding rms values during several preceding generations of a GA run. The network function, rms = $f(p)$, thus built up is able to calculate the rms corresponding to a specific parameter vector within milliseconds, whereas obtaining the same result by molecular mechanics methods takes several minutes for the problem at hand and with the equipment used. Therefore, significant time savings may be expected using a combination of GA optimization and NN simulation In addition, the simulated function, rms = $f(p)$, allows for insights into the dependence of the rms value on specific parameters or combinations thereof. Kohonen mapping is used as a tool to visualize such dependence.
 J Comput Chem 20: 455–471, 1999

Keywords: optimization of force field parameters; genetic algorithms; neural networks; Kohonen networks; backpropagation networks

Dedicated to Prof. Bernt Krebs on the occasion of his 60th birthday

Correspondence to: G. Huttner; e-mail: GA@indi.aci.uni-heidelberg.de

Contract/grant sponsor: Deutsche Forschungsgemeinschaft

Contract/grant sponsor: Höchstleistungsrechenzentrum Jülich

 CCC 0192-8651 / 99 / 040455-17

Introduction

Beyond the basic determination of reactivity by electronic factors, the reactivity of sizable molecules is very much determined by their conformational properties. Selectivity is in fact governed by this latter factor to a significant extent, and homogeneous catalysts owe their specificity mostly to steric factors.[1] It is a primary goal of chemistry therefore to understand the conformational properties of molecules.

Whereas quantum chemistry is, in principle, an efficient tool in this respect, its methods are not yet adequate to deal with conformational problems in the chemistry of complicated macromolecules (e.g., proteins or DNA) nor are they generally applicable to predict conformations of coordination compounds. This is especially true because coordination chemistry is, in essence, the chemistry of open-shell compounds, which are even more difficult to handle by quantum-chemical methods. In all cases in which quantum-chemical methods are either too time consuming to apply or not yet applicable, the methods of molecular mechanics may be the best alternative. These methods, first established to deal with problems in organic chemistry,[1,2] have now found wide and successful application in the chemistry of biomolecules and also, to some extent, in the field of coordination chemistry.[1–3]

Their widespread application in coordination chemistry is hampered by the fact that metals are highly variable with respect to coordination, and that force field descriptions of these various coordination forms are not generally available. On the other hand, these methods, if appropriately developed, would be of great help in predicting the conformational properties of coordination compounds. This would be of special value in those cases in which coordination compounds are used as catalysts.[1]

The traditional way to overcome part of these problems consists of selecting a specific class of compounds, to describe their geometry using a set of internal coordinates (e.g., distances, angles, and torsion angles) and to adjust the forces acting upon these variables using a trial-and-error process in comparing conformations calculated by the force field approach with molecular structures from diffraction data.[4,7] This type of parameter optimization necessarily means that only a minor part of the information available for a given class of compounds will be used, and that the estimates derived from analysis of this subset of information tend to be subject to the possibly biased estimate of the human analyst.

With this in mind we tried to develop methods by which the values of the relevant force field parameters are automatically generated by global optimization on the basis of the entirety of structural information available.[8] The procedure adopted is as follows:

1. Select a class of compounds and create a database containing all the structural information available for the individuals in this class.
2. Use statistical methods and pattern recognition techniques to find out whether the observed structures are not largely influenced by external forces within the crystals from which the structures have been determined.[9] If so, return to step 1. If not, proceed to step 3.
3. Globally optimize the relevant force field parameters so that the calculated structures are as close as possible to the observed ones. This is done by using the root-mean-square deviation (rms) between observed and calculated structures as the basis for evaluation of the fitness criterion in an optimization using a genetic algorithm (GA).[8]
4. Predict the outcome of experiments having results that can discriminate between ''correct'' and ''incorrect'' predictions of the force field model. Such validations include, for instance: predicting the structure of a compound[8] that had not been incorporated in the database; predicting NOE distances that are accessible to measurement; and predicting the energy of conformational isomerizations.[10,11]

The aforementioned procedure was applied to *tripod* metal compounds [*tripod* = $RC(CH_2X)(CH_2Y)(CH_2Z)$; X, Y, Z = $PR'R''$] and validation by experimental results lends quite some promise to it. One problem with this procedure involves the large amount of computer time necessary; the procedure must necessarily be performed in a parallel computing environment so as to produce results within a reasonable timeframe. There is yet another more intricate problem: With genetic algorithms automatically producing optimized parameter sets, it is difficult to obtain insights into

the specific significance of individual parameters. Such an insight would at least be intellectually appealing, but might also help to optimize the model subjected to optimization itself.

In this article we describe how the application of the methodology of neural networks (NN) may in part help to overcome the problems just discussed. It is shown that the thousands of locally optimized conformations accumulated during optimization by genetic algorithms form a basis upon which to train neural networks so as to reproduce the dependence of the quality of fit (rms) from the force field parameters. The neural network function thus built up may then be used to present the dependence between the parameters and the rms values in an intelligible way.

The rms predicted by this neural network function for a given set of parameters is found to agree with the one calculated for this same set of parameters by a full force field optimization procedure to within better than 5%. The rms values of parameter sets that had not been implemented in the training data set can also be predicted correctly. This means that the computationally simple NN function will adequately replace the computationally intricate process of evaluating an rms value by force field methods. This means that it should be possible to replace the time-consuming evaluation of the fitness criterion by force field methods during GA optimization by an appropriate NN simulation at different stages of the GA refinement process.

Description of the Chemical Problem: Definition of Parameters

By extended statistical analysis it had been shown that the conformations adopted by *tripod* metal templates (*tripod*M) in the solid state are not largely influenced by the forces acting upon them within the crystal, and thus represent innermolecular forces.[9] This means that the conformations observed correspond to minima on the molecular energy hypersurface. A necessary condition for a force field description of these templates is therefore that the force field has to reproduce these conformations as at least local minima on the hypersurface generated by the force field. This means that, in setting up a force field description of these templates, the force field parameters must be chosen such that this necessary condition is fulfilled to the extent possible. Therefore, the problem is finding an appropriate set of parameters subject to this condition. Mathematically, this is a problem of global optimization. Among the few methods available for tackling problems of this type, GA optimization[12,13] has been selected, and has previously been shown to be successful in deriving a force field description of compounds *tripod*$Mo(CO)_3$, which correctly predict structures,[8] NOE distances,[10] and isomerization barriers.[11] As this article deals with the analysis and the modification of the GA refinement process for the same class of compounds, the basic structure of the compounds, the members present in the database and the force field parameters used to describe them are given in Table I and Figure 1.

All the compounds contain a $Mo(CO)_3$ entity that is η^3 coordinated to a *tripod* ligand. The geometry of the framework is thus alike for all compounds in the sample and its basic features are shown in Figure 1 for $CH_3C(CH_2PhEt)_3Mo(CO)_3$ as an example. The force field description of compounds given in Table I was set up as follows: all terms not involving the metal were taken from the well-established mm2* set of parameters.[14] Terms involving immediate contributions by the metal were defined as indicated in the legend of Figure 1. From the total of 16 variables in the terms thus defined, only 14 were allowed to refine. The force constant, k_{Mo-C1}, was set to the fixed value of 2.0 mdyn $\cdot$ Å^{-1}. The equilibrium angles, $\alpha_{P-Mo-C1}$, were set to the corresponding mean value of the

TABLE I.
Constitution of the Ten Compounds Comprising the Database.

Compound	Formula
1	$CH_3C(CH_2PPh_2)_3Mo(CO)_3$
2	$CH_3C(CH_2PBzlPh)_3Mo(CO)_3$
3	$CH_3C(CH_2PMe_2)_3Mo(CO)_3$
4	$PhCOOCH_2C(CH_2PPh_2)_3Mo(CO)_3$
5	$CH_3C(CH_2PNap_2)_3Mo(CO)_3$
6*	$CH_3C(CH_2PEtPH)_3Mo(CO)_3$
7*	$CH_3C(CH_2PEtPh)_3Mo(CO)_3$
8	$ClCH_2C(CH_2PPh_2)_3Mo(CO)_3$
9	$PhCH_2C(CH_2P(DBP))_3Mo(CO)_3$
10	$CH_3C(CH_2PEt_2)_3Mo(CO)_3$

Abbreviations: Bzl: benzyl; DBP: dibenzophospholyl; Et: ethyl; Nap: naphtyl; Ph: phenyl. Compounds **6** and **7**, labeled with an asterisk, are two crystallographically independent conformations of $CH_3C(CH_2PEt)_3Mo(CO)_3$, as found in one and the same unit cell.

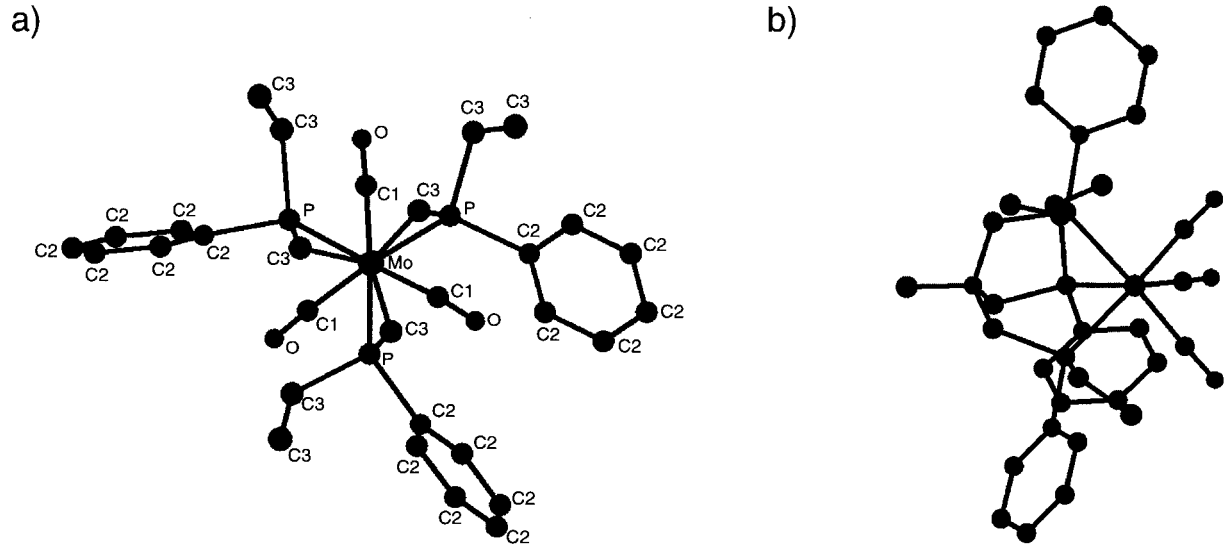

FIGURE 1. Constitution and conformational characteristics of the molecules in the database illustrated for $CH_3C(CH_2PhEt)_3Mo(CO)_3$ (**6**, Table I) in two projections. The atom designators given in (a) refer to the terms used in the force field description in each case. The parameters involving a contribution by the metal had been refined; for example, α_0 and k_a: C1 — Mo — C1, P —Mo — P, Mo — P — C3, etc.; r_0 and k_b: Mo — P, Mo — C1.[8]

sample; that is, to 95.6° and 176°, respectively, and held fixed.[8]

Description of the Logical Structure of Computational Methods Applied

GENETIC ALGORITHMS

Optimization by Genetic Algorithms works as follows (Figure 2): For each variable parameter define a range within which it will be allowed to vary. Define the resolution within this range. Define a rule to uniquely encode values within this range as binary numbers. Define a binary string to hold the binary coded parameter values at definite positions of this string. The whole parameter vector comprising the n parameters as components is thus represented by a binary string the length of which depends on the number of parameters and the resolution chosen for the individual parameters. Define a function which produces a number which bears a relation to the quality of fit obtained by a given vector of parameters.

In the problem at hand, this function is taken as the rms deviation in the positions of all atoms over the entire sample of ten compounds (Table I). It is evaluated by adjusting the observed and calculated structures of the individual molecules with respect to their mutual translation and rotation such that the rms is minimal for each molecule. Calculating the structures means optimizing the structures by molecular mechanics methods with the observed structures as the starting point. Now the problem is prepared for GA optimization:

Within the constraints just discussed, parameter vectors are generated at random and produce a first generation of a ''population,'' containing randomly generated binary string representations of p such vectors (Fig. 2).

The fitness of these parameter vectors is evaluated; that is, they are optimized by the molecular mechanics approach, and the rms between the observed and calculated positions is determined and encoded as a fitness value such that the higher the fitness, the larger the number (Fig. 2).

From now on apply the two basic biomimetic optimization strategies:

1. Let the members of the population create offspring with a probability proportional to their fitness. The offspring is generated by randomly selecting a position in the binary strings of the parents where crossover takes place (Fig. 2, bottom). The expectation that this process will generally lead to a higher overall fitness of the offspring generation relies on the assumption that those bitstrings which had already been found to be efficient will predominately be inherited.

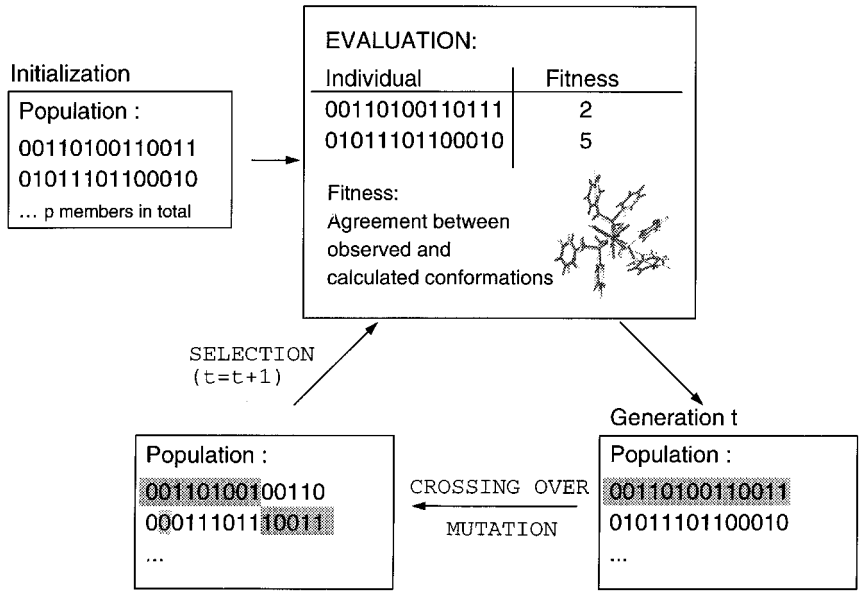

FIGURE 2. Schematic representation of GA optimization of force field parameters.

2. Occasionally allow for mutation, i.e. allow for individual bits within the parameter strings to change from 0 to 1 and vice versa (Fig. 2, bottom), so as to warrant sufficient diversity in the ''genetic pool.''

The process is repeated over many generations and generally found to converge, producing some optimal set of parameters. There is no guarantee that *the* optimal set will be found in a finite number of generations nor is there, with the problem at hand, a compelling reason to assume that only one single optimal parameter vector might exist.[8] Therefore, the only physical meaning of an optimal parameter vector is that it reproduces the observed structures to the extent possible when it is used in a molecular mechanics optimization procedure. The aim of this work is not to show that GA optimization of force field parameters is an efficient tool—that conclusion has already been reported.[8] The primary objective is rather to make use of the immense amount of information that accumulates as a necessary byproduct during GA optimization by the evaluation of so many parameter vectors with respect to the rms they produce. It will be shown how this information may be used to analyze and optimize the refinement process itself. NNs are used as an efficient tool to this end.

SELF-ORGANIZING MAPS

Neural networks are an efficient tool in pattern recognition.[15,16] Given a set of data they are able to develop classification criteria for these data on their own and to classify the data according to these criteria. As with all pattern recognition techniques, the definition of similarity is at the basis of their logic. The Euclidean definition of distance is used as a measure throughout this article, which is the case for most applications of NNs; that is: the more distant the objects are with respect to their coordinates in n-dimensional space, the less similar they are. To explain the basic functioning of pattern recognition by NNs, Figure 3 shows the basic logic of the type of network used in this analysis.

The network shown belongs to a class of networks called self-organizing nets that had first been proposed by Kohonen.[17] This type of network is capable of grouping objects according to their similarity by means of a projection, which preserves their topological relations. As with all pattern recognition techniques it relies on creating an informative projection from the multidimensional data space to a low-dimensional analysis space such that relations between the data items are immediately intelligible to the human perception apparatus. To explain how this type of network

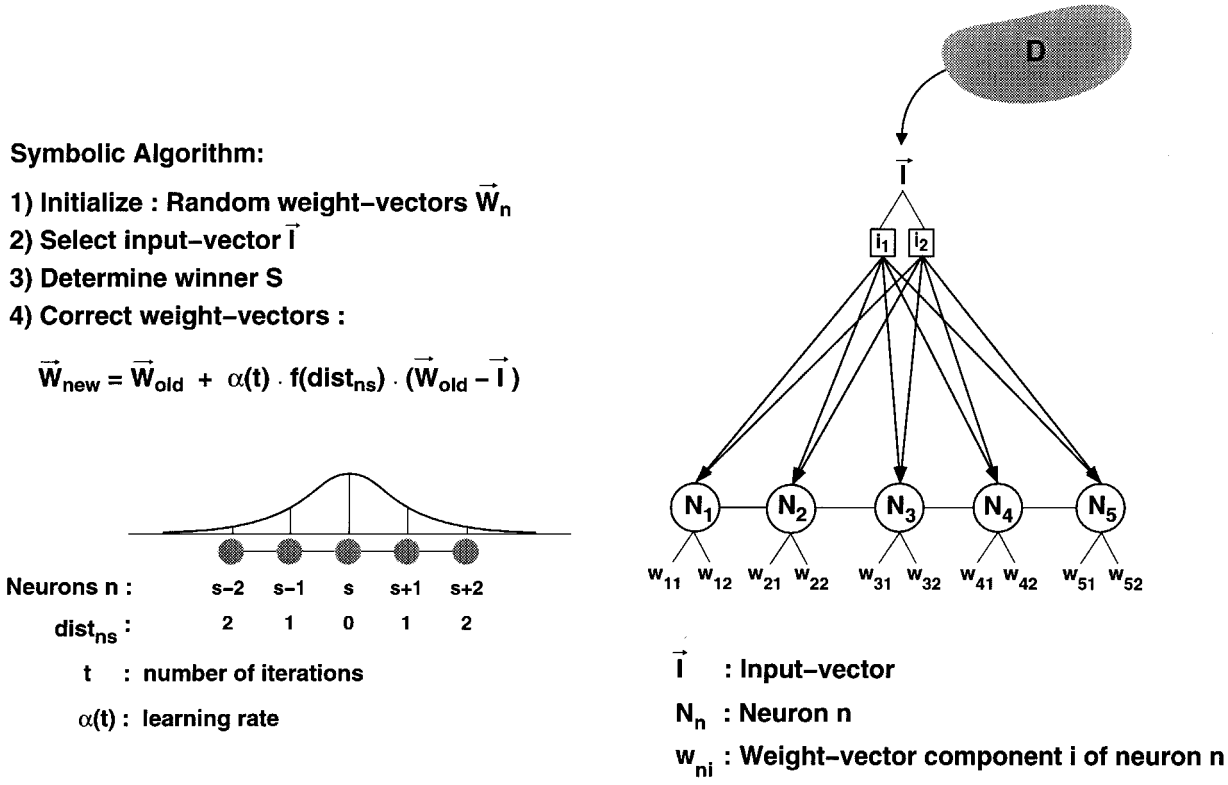

FIGURE 3. Algorithmic structure of a self-organizing net.

operates, imagine a two-dimensional data space, **D** (Fig. 3), with each of the objects in this space characterized by two coordinates. The coordinates of the data points in **D** may be presented to the network by the input units i_1, i_2. At the bottom of the diagram (Fig. 3), five items labeled N1–N5 are shown, which are called "neurons" in the appropriate language. Each of these neurons stores two coordinates with components, w_{n1} and w_{n2}, which are called the "weight vectors" of neuron N_n. Analysis of the data in **D** by the aforementioned network proceeds as follows:

1. Initialize the weight vectors at random.
2. Select a data point from **D** at random and present it to the network.
3. Find the neuron that has a weight vector closest in value to the input vector **I**. This neuron is called the "winner neuron" (Fig. 3).
4. Change the weight vector of the winner neuron so as to become closer to the input vector, and do the same type of adjustment for the neurons in the neighborhood of the winner neuron, albeit reducing the extent of adjustment with increasing distance from the winner neuron (Fig. 3). Repeat steps 2–4 until convergence is obtained (i.e., until the components of the weight vectors, w, no longer change to a significant extent.

It has been found that this process will generally converge such that the neurons represent a set of converged weight vectors. The logic of the process means that a sorting machine has been built up, which, by comparison of data vectors with weight vectors, allows the data points to be sorted to the different neurons. Objects that closely resemble one another in data space will be assigned to the same neuron. At the same time, the procedure for correcting the weight vectors of neurons in the neighborhood of the winner neuron preserves the topology of the data space in projecting data close in data space if not onto the same neuron so at least onto neurons in close neighborhood. The power of the method will become apparent if the data space is of higher dimension and therefore if the data space itself is not immediately

accessible to intelligible visualization. A projection of such higher dimensional data onto a low-dimensional and thus intelligible visualization space would be of great help in analyzing the data for deciphering patterns hidden in the data and only immediately apparent in their projection. The strategy described here may be designed analogously for a two-dimensional projection space with the basic meaning staying the same: Neurons close to each other in the projection plane represent objects similar to each other in data space. The weight vectors associated with each neuron represent objects in data space that are very similar to one another. This type of network has been used in this work to project a 14-dimensional space of force field parameters onto a two-dimensional plane (discussed later).

BACKPROPAGATION NETWORKS

Yet another type of NN has been used to simulate the quality of fit obtainable with a given set of force field parameters, p. As a measure of the quality of fit, the rms deviation between observed and calculated structures, as given earlier, is used. This complicated function, rms $= f(p)$, can only be defined in operational terms. It may be simulated by a NN if sufficient points of the hypersurface, rms $= f(p)$, are known. The basic architecture of a NN capable of this type of simulation is shown in Figure 4.

The network is composed of layers of operational entities (i.e., neurons) with the neurons in every pair of such layers fully connected (Fig. 4b). In the diagram, the output of a neuron from a layer on the left-hand side serves as input for the right-hand-side neurons. The output of a left-hand side neuron, however, is not used as such, but rather is multiplied by the weight, w (Fig. 4), assigned to the specific connection, before it serves as an input to the neuron in the next layer. These weights are often called ''synaptic strengths'' as in biological neural networks.

Each of the neurons on the right-hand side of the input layer is an operational unit performing a function (as shown in Fig. 4a): The input (`In`) is calculated as a weighted sum of the output signals of the neurons on its left-hand side. The sum is taken over all connections linking the given neuron with the neurons on its input side. The output of the neuron (`Out`) is limited between 0 and 1 with small values of `In` producing the response of

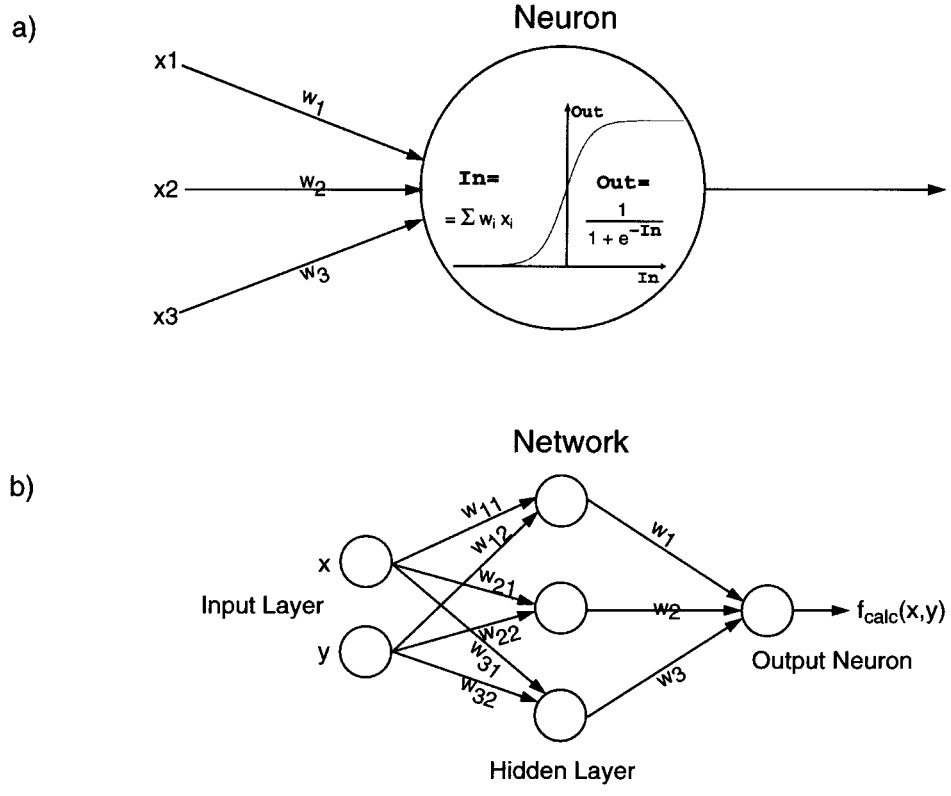

FIGURE 4. Illustration of the basic architecture of a backpropagation network.

0, and large values of In resulting in an output of 1. Intermediate values of In will produce an output between these extreme values. The function used to transform In into Out is sigmoidal:

$$\left[\text{Out} = \frac{1}{(1 + e^{-\text{In}})}\right]$$

A network as described (Fig. 4b) will produce an output, $f(x,y)$, for every pair (x/y) fed into it via the input layer. Given the architecture of the network, $f(x,y)$ is determined by the weights, w.

Networks of this type may be used to simulate function $f(x,y)$ if the values of this function are known for a sufficient number of (x/y) pairs that must cover the subspace in which the function is to be simulated. Simulation proceeds as follows: The weights, w, are initialized at random. The (x/y) pairs are fed into the network, one after the other. The result, f_{calc}, produced at the output neuron is compared with the exact value, f_{obs}, which the function should have at a given (x/y) pair (scaling of f_{obs} into the range of $0 \leq f_{\text{obs}} \leq 1$ is of course necessary to make these items comparable). The sum of squares of the deviations, $f_{\text{obs}} - f_{\text{calc}}$, is accumulated over all the (x/y) pairs contained in the training set. The weights, w, are then adjusted so as to minimize this sum. If gradient methods are used for this purpose, correction of the weights must be done starting from the output side toward the input side (first w_i and then w_{ij}; see Fig. 4b). For this reason, these algorithms are often called ''backpropagation'' networks. Due to the nonlinearity of the problem, the process must be repeated until convergence is obtained.

To illustrate the potential of this type of simulation procedure, Figure 5 provides an example: Using the network architecture shown in Figure 5b, the function $f(x, y) = z = \sin x + \cos y$ has been simulated in the range of $0 \leq x, y \leq 2.2\pi$ based on a training set comprising the values on an evenly spaced 15×15 grid. It is seen that the weights assigned to the different connections have

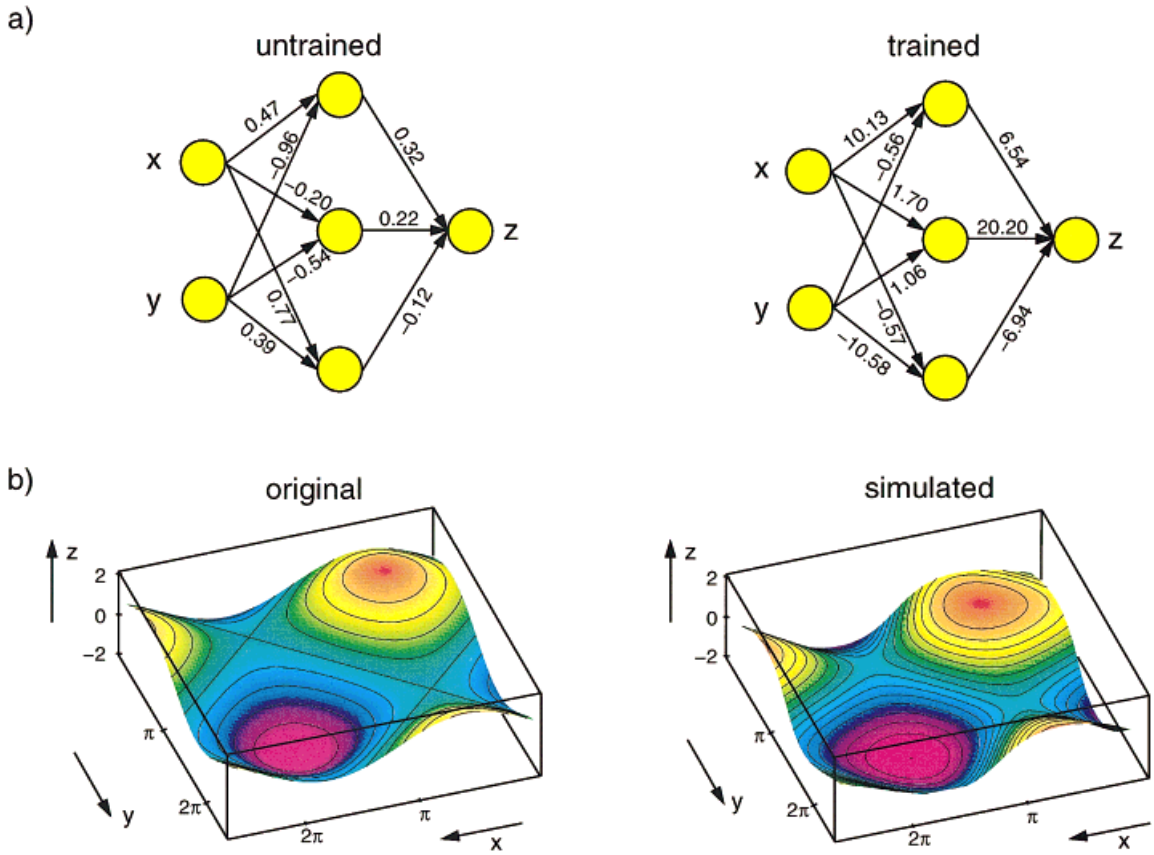

FIGURE 5. Simulation by neural networks. Illustration of the simulation of the function, $z = \sin x + \cos y$, by a simple backpropagation network. The numbers in (a) assorted to the connections of the network represent the values of the weights, w, as defined in Figure 4. Figure 5b shows a color-coded contour diagram of the function, $z = \sin x + \cos y$, as the original on the left-hand side and its simulation by the trained network on the right-hand side.

been drastically changed from their random values characterizing the untrained network to the specific values characterizing the trained one (Fig. 5a). The quality of simulation obtained by even this simple type of network is apparent from Figure 5b. For more complicated functions more intricate networks with more than one hidden layer may be used. The basic logic of the simulation procedure will stay the same, independent of the specific architecture.

In the present work, this type of simulation procedure has been used to model the function rms = $f(p)$. The strategy is applied to mimic the dependence between force field parameters and rms values as calculated by force field methods during the many iterative runs of GA optimization of these parameters.

Exploration of Parameter Space by Self-Organizing Maps

Optimization of force field parameters by genetic algorithms necessarily provides evaluation of thousands of parameter vectors with respect to the optimal rms value they are able to produce. With respect to the final goal of the optimization procedure (i.e., finding the best parameter vector) these intermediate evaluations are no more than a necessary part of the process with no immediate significance in the final result. On the other hand, much information is accumulated during the GA process; that is, each parameter vector for which the rms value has been determined during a GA run marks a point on the hypersurface, rms = $f(p)$. In general, when applying GA to any kind of problem, no further use of this huge amount of data is made, even though it is available at no additional expense. It is evident, however, that, whenever the function for the evaluation of a given parameter vector is complicated, this information will be helpful to get obtain insight into the characteristics of functional dependence.

In the present case, the evaluation of the rms for a given parameter vector is a complicated function indeed, as the rms is the result of a multistep gradient refinement itself. The evaluated parameter vectors themselves represent points in n-space ($n = 14$ for the example analyzed); to extract regularities from this sample it is necessary to find an appropriate projection onto a lower dimensional and thus immediately intelligible space ($n = 1, 2, 3$). In the present case, a Kohonen network (see earlier) was used to generate a 2D projection of the sample. By this type of mapping, the topological relations between the points in n-space can be mapped correctly. This means: areas close to each other in the 2D map represent points in n-space that are close to each other. Or, in other words: parameter vectors that are similar to one another will be mapped in close proximity and are thus classified according to their similarity.

To visualize the relation between the parameter vectors and their respective rms, the rms can be computed for each class. This is done by taking the mean of all the rms values corresponding to all vectors in this class. The map is then colored according to these rms mean values. Figure 6 shows the result of this procedure for the case in which 19,798 parameter vectors, representing 14 parameters each, were projected onto a Kohonen map of 50×50 classes.

The right-hand side of Figure 6 represents this map at an early stage of self-organization (i.e., based on a random selection of weight vectors). The color coding by the corresponding rms values produces a random pattern, as expected (the areas remaining colorless are those with no representative in data space).

The left-hand side of Figure 6 shows the result after convergence of the self-organizing process. Color coding reveals a clear structure with a concentration of high rms values of about 0.43 Å (green) around class (30/30). This means that there is a wide range of classes of parameters that tend to give low rms values (red, orange) and a relatively minor range connected with high rms values (yellow, green). It appears therefore that many classes of parameter vectors do not greatly influence the quality of fit, whereas, conversely, there is a restricted set of variations that do so strongly.

To find out which variations are the ones that give rise to a poor quality of fit, the map is colored according to typical values that individual parameters have in each class; that is, according to individual components of the weight vectors. It is expected that parameters connected with a poor quality of fit will produce color patterns similar in shape to the color pattern observed by coloring the map on the basis of the rms values. Parameters not directly connected to the local degree of quality of fit should produce patterns that are dissimilar to the one shown in Figure 6a. Such plots have been generated for all 14 components of the parameter vectors. Figure 7 shows two illustrative examples.

It is seen that force constant, $k_{\mathrm{Mo—P—C3}}$ (c.f. Fig. 1), determining the stiffness of the corresponding

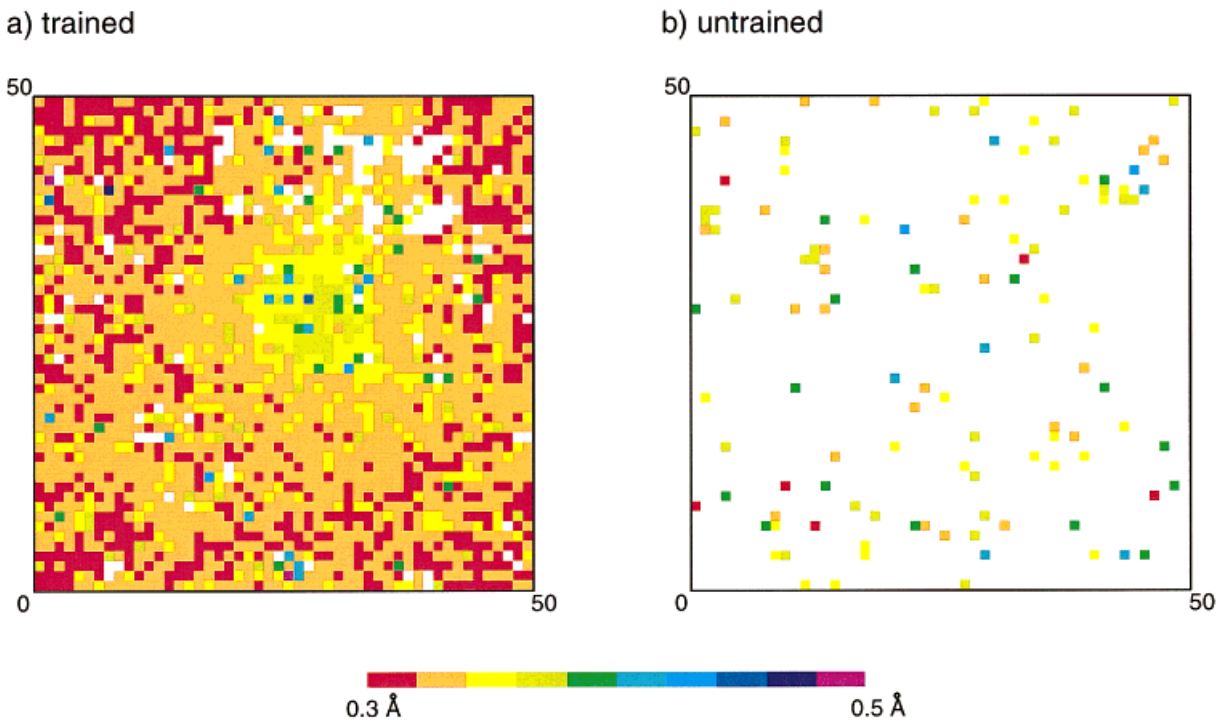

FIGURE 6. Projection of 19,798 parameter vectors from 14-dimensional space onto a two-dimensional 50 × 50 Kohonen map. The 50 × 50 classes of parameters thus defined are color coded by the quality of fit they are able to produce (rms). The right-hand side (b) illustrates the distribution observed at the start of the self-organizing process. The left-hand side (a) represents the distribution obtained after convergence. The map is of the toroidal type.

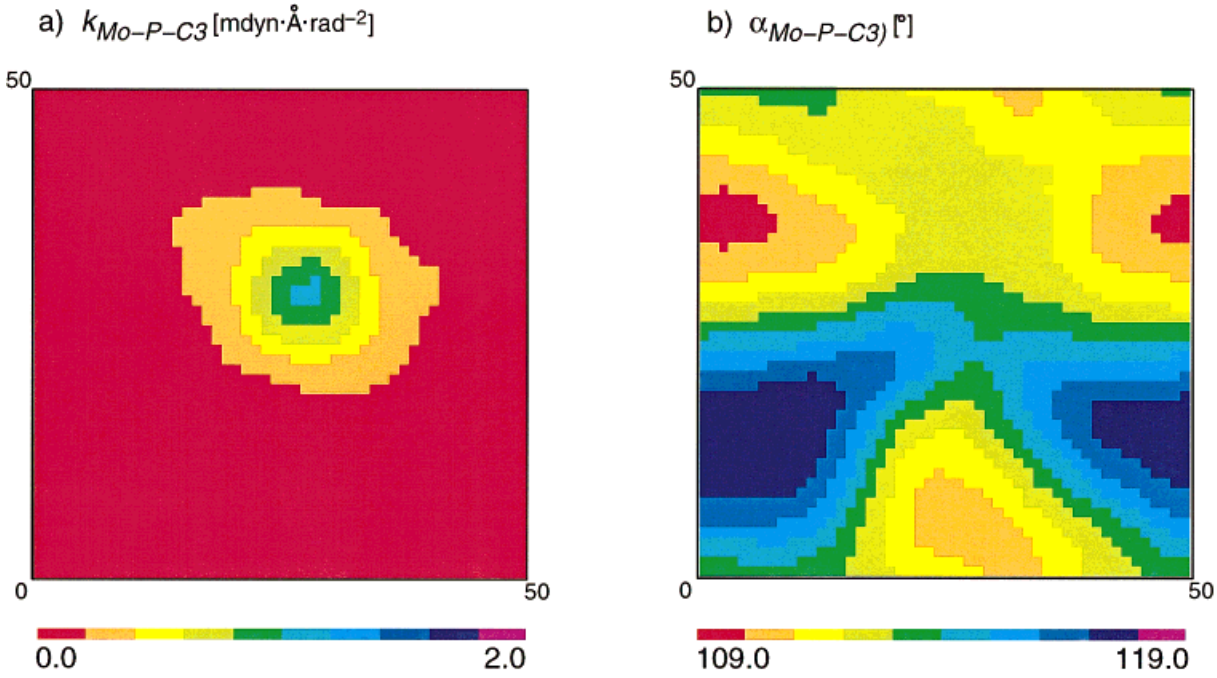

FIGURE 7. Kohonen map as in Figure 6a color coded by the values of specific components of the weight vectors. The pattern produced by color coding on the basis of $k_{Mo—P—C3}$ (a) shows a strong resemblance to the one in Figure 6a, whereas color coding by $\alpha_{Mo—P—C3}$ produces a distinctly different pattern.

bending motion has large values in just the same area that marks parameter vectors of low quality in Figure 6a. This means that parameter vectors that tend to give a poor quality of fit will also tend to have their component representing $k_{Mo—P—C3}$ at a high value. In Figure 7b, the pattern obtained by coloring the map according to the corresponding equilibrium bond angle, $\alpha_{Mo—P—C3}$ (c.f. Fig. 1), shows no obvious resemblance to the pattern shown in Figure 6a. This means that there is no direct connection between this parameter and the quality of fit. From Figure 7a, it appears that, as long as the corresponding force constant, k, is small enough (Fig. 7a, red area), there are many good solutions to the problem (compare the large number of good solutions [red in Figure 6a] in the corresponding area). Comparing Figure 7b with Figure 6a shows that good solutions may be obtained for a broad range of equilibrium angles, $\alpha_{Mo—P—C3}$.

This type of easy-to-perform analysis gives some insight into the dependence of the quality of fit on the ranges of values of individual parameters, and may therefore help to speed up the optimization process by excluding parameter ranges that give a poor quality of fit at some intermediate stage of refinement. In addition to this practical importance, this type of analysis is appealing because it allows insight into the influence of individual parameters on the quality of fit that would otherwise be cumbersome to obtain.

Simulation of Force Field Calculations by Backpropagation Networks

If a backpropagation network of suitable architecture is fed with the parameter vectors accumulated during part of a GA optimization, and is trained with the corresponding rms values also available from GA optimization, it should be capable of representing the function, rms = $f(p)$. The kind of network used (see earlier) in the case analyzed is composed as follows:

The input array presenting the 14 components of the parameter vectors is connected to a first hidden layer of 14×14 neurons, which is again connected to a 7×7 hidden layer. The information then passes a final 4×4 hidden layer to produce the output value (rms) in the final entity.

During the training process, the network learns to simulate the function, rms = $f(p)$, by adopting the weights associated with each neuron. The efficiency of this learning process is shown in Figure 8, where the sum of the squares of the errors:

$$\sum (\mathrm{rms}_{obs} - \mathrm{rms}_{calc})^2$$

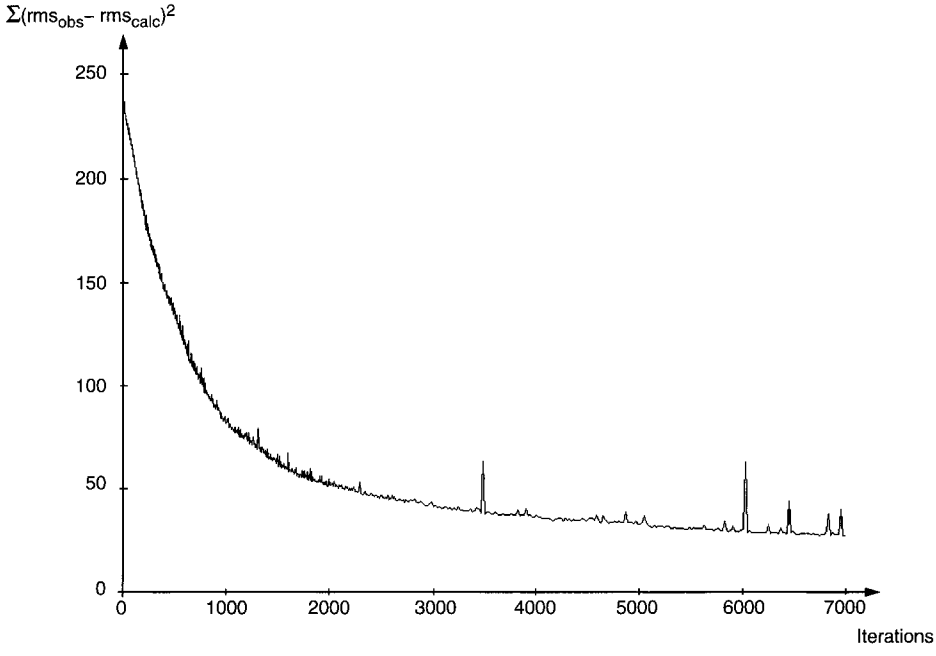

FIGURE 8. The process of learning to simulate the function rms = $f(p)$ by backpropagation. A backpropagation network as defined in the text was trained on the basis of the rms values known for 19,798 parameter vectors p.

over the whole set of the training data (19,798 parameter vectors) is plotted against the number of training cycles.

It is seen that this sum converges to an ordinate value of around 30, with a nearly exponential increasing quality. The occasional spikes in the curve are characteristic of the training protocol used here (see later). At the limit of numerical convergence, the rms deviation between the calculated and the given rms values is only $7 \cdot 10^{-3}$ Å. This means the network has learned to reproduce the rms values from the parameter vectors with which it was fed.

The question is then whether the trained network will be able to predict rms values for parameter vectors that had not been used in training. Whereas it cannot be expected that the network could predict rms values for parameter vectors, the components of which lie outside the range representative of the training set of vectors, it can well be expected that it will interpolate within the range it was trained. As a way to validate this expectation, the parameters were projected on a 30×30 Kohonen map.

In Figure 9a, the map was colored as described previously according to the average rms for each class. In Figure 9b, the same map was used and colored by the rms computed by the trained network for each characteristic weight vector. The patterns obtained by these two procedures are very similar. The difference in appearance results from the fact that mean rms values for each class were used in Figure 9a, whereas the rms value calculated for the mean vector of each class was used to color code the diagram in Figure 9b. Therefore, the averaging process is less smooth in Figure 9b. The overall features are nevertheless identical for both maps.

The trained network may now be used to analyze projections of the parameter space in more detail. Setting the values of 12 parameters from the total of 14 to the optimal values found by GA optimization, the remaining two parameters may be varied in the range that was allotted to them for refinement in a systematic way, and thus a map relating pairs of values of these varied parameters to the rms they produce may be constructed.

In contrast to the information obtained from Figures 6 and 7, which pinpoint the regions in parameter space that will lead to a poor quality of fit, those regions can now be analyzed that embed good parameter values, in the sense that they are linked with low rms values. Figure 10a shows the relevant part of a diagram linking the rms value with the force constant, $k_{\mathrm{Mo—P—C3}}$, and the equilibrium angle, $\alpha_{\mathrm{Mo—P—C3}}$, of a Mo—P—$C_{sp3}$ bending motion.

It is seen that only low values of the relevant force constant will produce low rms values (as already apparent from Fig. 7a). If the value of this force constant is too high, then the obtainable agreement is poor for the whole range of equilibrium angles (e.g., $k \geq 0.125$ mdyn $\cdot$ Å $\cdot$ rad^{-2}). If, on the other hand, the force constant is low enough, optimal ranges for the equilibrium angle become apparent.

Analyzing the relation between the two equilibrium angles Mo—P—C_{sp3} and Mo—P—C_{sp2} ($\alpha_{\mathrm{Mo—P—C3}}$, $\alpha_{\mathrm{Mo—P—C2}}$) in the same way, the diagram shown in Figure 10b is obtained. It is seen that there is a narrow range for values of $\alpha_{\mathrm{Mo—P—C2}}$ that will induce a good quality of fit, whereas the quality of fit is not strongly dependent on $\alpha_{\mathrm{Mo—P—C3}}$ as has already been inferred from the analysis of Figure 7b. The optimum shown in Figure 10b at $\alpha_{\mathrm{MoPC2}}/\alpha_{\mathrm{MoPC3}} = 117/110°$ is a local optimum. The global optimum, as found by GA optimization, is at 119/116°, close to the second minimum shown in Figure 10b. If the section at $\alpha_{\mathrm{Mo—P—C2}} = 119°$ of Figure 10b is analyzed in more detail, the variation of the rms value against $\alpha_{\mathrm{Mo—P—C3}}$ reveals, as already indicated by Figure 10b, that there are two minima on this curve separated by a maximum at around 114°.

In trying to set up an optimal force field description for compounds *tripod*$Mo(CO)_3$, the equilibrium angles, if not refined, should therefore be set close to the relevant optimal values. Taking the average of all observed Mo—P—C3 angles as the basis should lead to a reduced quality of agreement, as this average value is 114.5°, and thus close to the maximum of the curve represented in Figure 11. On the other hand, the rms differences due to variations in these angles are not very large (Fig. 11), and it may be warranted, for the sake of simplicity, to use an average angle in the force field description as has in fact been done in another study.[8]

GA Refinement by Calculating the Fitness Criterion on the Basis of NN Simulations

As has been shown, a trained backpropagation network will correctly predict the rms value that would be obtained based on a given set of parame-

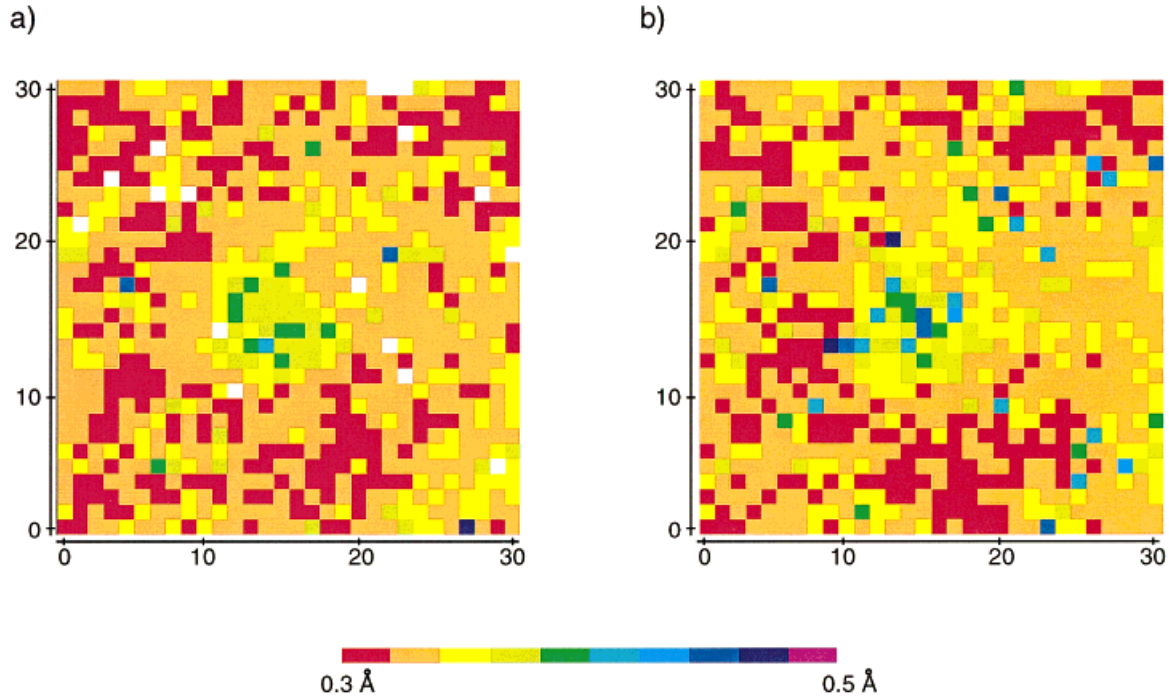

FIGURE 9. Classification of 19,798 parameter vectors by a 30 × 30 Kohonen map. Color coding by rms values. (a) rms calculated by molecular mechanics methods. (b) rms calculated by a trained backpropagation network. When comparing (a) with Figure 6a, which shows a map of higher resolution, the toroidal topology of these maps has to be taken into account — shifts of the patterns as a whole in any direction have no physical importance.

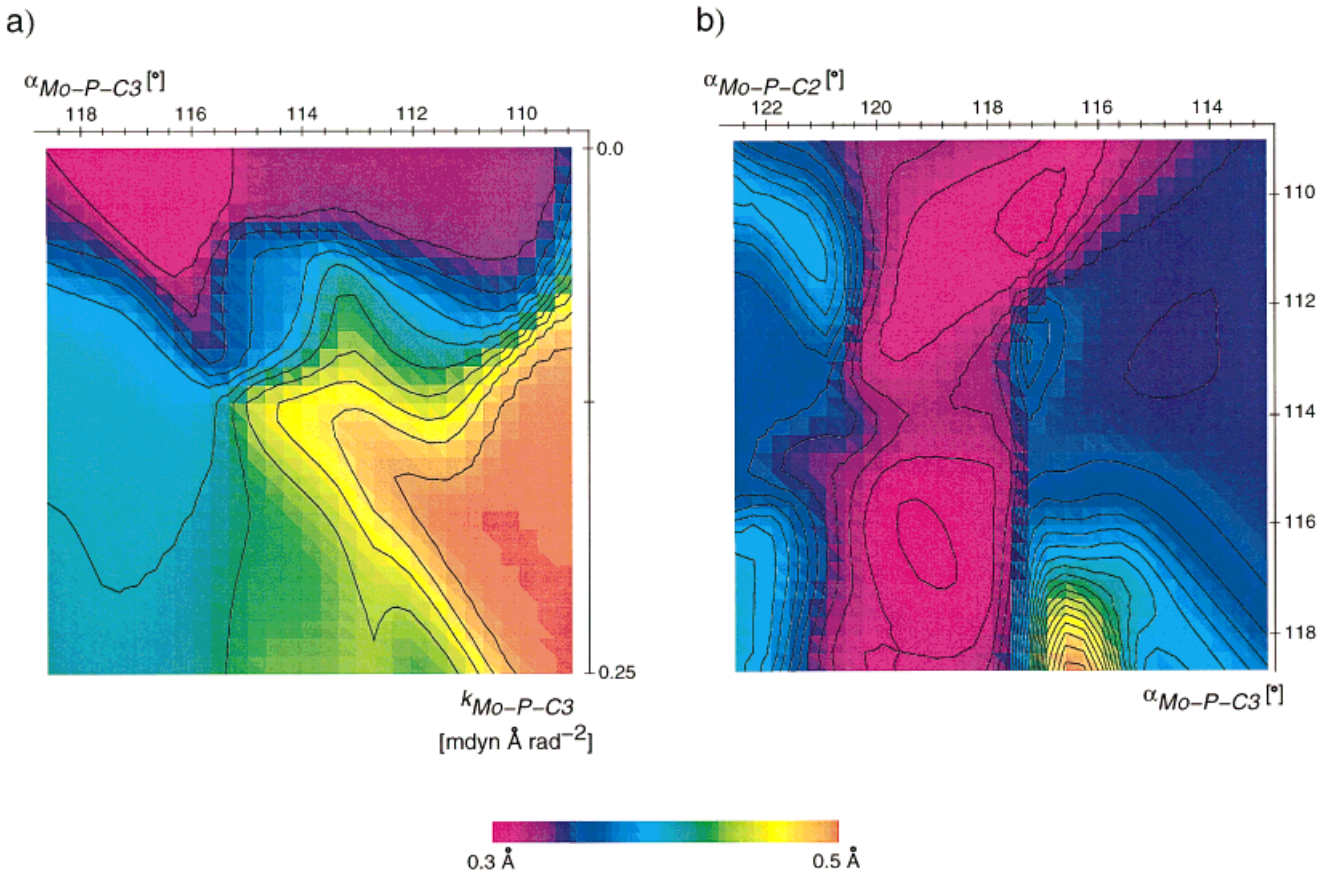

FIGURE 10. Selected examples of the dependence of the rms value on individual combinations of force field parameters. Color coding with respect to rms value calculated by the trained network.

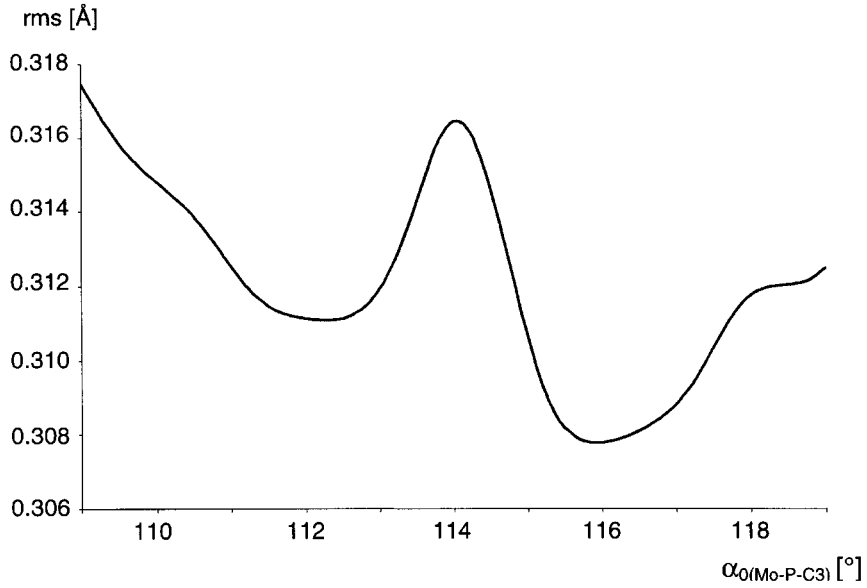

FIGURE 11. Dependence of the rms value on the variation of a specific equilibrium angle. The values of the remaining 13 parameters of the problem were set to their optimum.

ter values by a complete force field optimization procedure. Whereas computing this rms value by molecular mechanics methods for a set of ten molecules will take around 3 minutes with the equipment used (see later), the network will evaluate this rms value in less than 1 millisecond. Because the time-consuming step in the GA optimization of force field parameters is just this evaluation of rms values, the process could be speeded-up by several orders of magnitude whenever a trained neural network replaces the time-consuming molecular mechanics calculations. It is shown in this section that it is in fact possible to substitute the molecular mechanics calculation during a GA run by NN estimation. Replacing the molecular mechanics evaluation by the trained neural network (see previous section) leads to optimized parameters with almost the same predictive quality as those obtained using the standard molecular mechanics evaluation tool. Computation time, on the other hand, represents only a minor fraction (see next section). The behavior of the refinement process is equivalent to the one observed when fitness is evaluated by MM methods (Fig.12).[8]

The best parameter set obtained, if used in a molecular mechanics evaluation, leads to an rms of 0.315 Å over the whole sample (the estimate by the neural network function of rms = 0.303 Å [see Fig. 12] is only slightly different). Refinement using the molecular mechanics evaluator leads to an rms of 0.306 Å. The rms deviations for the individual compounds within the data set, as obtained by the two types of refinement (mm2*n*: fitness evaluation by neural network; mm2*m*: fitness evaluation molecular mechanics), are shown in Table II and are found to be closely similar.

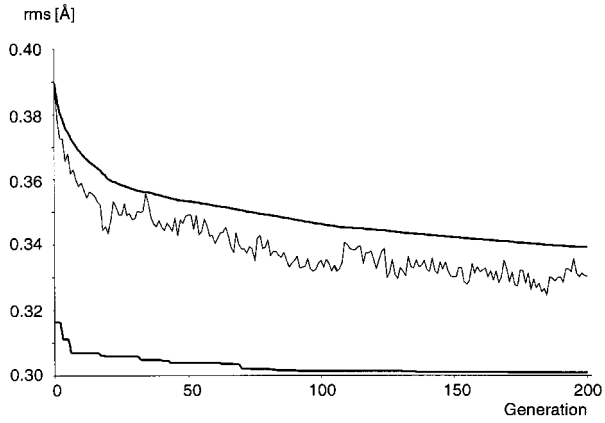

FIGURE 12. Convergence of GA optimization of force field parameters with a trained backpropagation network used as the tool to evaluate the fitness (mutation probability: 0.02; crossing over probability: 0.9). First trace (top): mean value of rms values obtained at all preceding generations, including the value of the specific generation considered; second trace (middle): mean rms for each generation; third trace (bottom): rms value of the best individual in each generation.

TABLE II.
Comparison of rms Values.

Compound	rms_{mm2n}[Å]	rms_{mm2a}[Å]
1	0.262	0.289
2	0.470	0.422
3	0.224	0.201
4	0.580	0.584
5	0.361	0.329
6	0.250	0.256
7	0.208	0.195
8	0.231	0.234
9	0.309	0.303
10	0.252	0.249
rms_{mean}	0.315	0.306

rms_{mm2n}: GA refinement of force field parameters based on evaluating the fitness by a trained backpropagation network; rms_{mm2a}: GA refinement of force field parameters based on evaluating the fitness by molecular mechanics. rms values for the individual compounds as obtained by molecular mechanics calculation with the optimal parameter set in each case.

The procedure described here clearly cannot entirely replace the time-consuming molecular mechanics evaluations necessary for optimizing force field parameters on the basis of molecular structures. However, it is conceivable that, at a stage of refinement at which a few thousand evaluated parameter sets have been generated, this information may be used to train a network and to then replace the molecular mechanics evaluator by the NN simulation and thus permitting the process to converge very quickly. At this stage, another process using the molecular mechanics evaluator is started and the whole procedure repeated until convergence.

Whereas the saving in computing time as far as the evaluation of rms values itself is concerned will be on the order of 10^5, the time-consuming step of training the network will counterbalance these time savings to a certain extent. The optimal strategy in applying this methodology has yet to be found.

Computational Details

NEURAL NETWORK SIMULATIONS

Self-organizing networks. The KMAP program from J. Gasteiger et al. has been used to perform self-organizing network analysis and to produce the color-coded Kohonen maps shown in Figures 6, 7, and 8. In this program, the following neighborhood function, *fdist*, is implemented:

$$fdist(dist_{ns}, r) = \begin{cases} 1 - \dfrac{dist_{ns}}{r} & \text{if } dist_{ns} < r \\ 0 & \text{else} \end{cases}$$

where r is the neighborhood radius and $dist_{ns}$ is the distance between winner neurons and neuron n (see Fig. 3). The neighborhood radius, r, is reduced iteratively every n_r learning steps by a value of λ_r. Also, the learning rate, α (see Fig. 3), is reduced during the training process. This is done by multiplying α every n_α iterations by a factor λ_α lying between 0 and 1.

The topology of the Kohonen network used is toroidal, which means that a repetitive periodic pattern, obtained by shifting the original pattern along the axes over the whole length, can be generated in the same way as the structure of a crystal is represented by the contents of its unit cell. In other words, at a given ordinate value, the rightmost pixel represents the pixel at the immediate left of the leftmost pixel of the same ordinate value at the same time. The leftmost pixel represents the immediate neighbor of the rightmost pixel at this ordinate value. The same type of cyclic neighborhood relation applies in the other direction as well.

The training parameters for the 50×50 and 30×30 Kohonen maps shown in Figures 6 and 8 are as follows:

- 50×50 network: number of iterations (i.e., the number of parameter vectors presented to the network): 500,000; $\alpha_0 = 1.0$; $r_0 = 25$; n_r, $n_\alpha = 4000$; $\lambda_r = 0.2$; $\lambda_\alpha = 0.95$.
- 30×30 network: number of iterations (i.e., the number of parameter vectors presented to the network): 250,000; $\alpha_0 = 0.7$; $r_0 = 10$; n_r, $n_\alpha = 4000$; $\lambda_r = 0.16$; $\lambda_\alpha = 0.7$.

Backpropagation networks. For the backpropagation network analysis the neural network program package, Stuttgarter Neural Network Simulator (SNNS, Version 4.1), was used.[18] Training was done using the standard backpropagation algorithm (Std_Backpropagation) with the learning rate, $\alpha = 0.8$, and the activation function as previously expressed and also in Figure 4. In this algorithm, correction of weights is performed according to a simple steepest descent gradient minimization procedure. Weights have been initialized randomly.

In the SNNS control panel, the following options have been set:

```
Learning func: Std_Backpropagation
Update func: Topological_Order
Init func: Randomize_Weights
Cycles: Shuffle
Init:1    −1
```

Training data set. The training data set contained 19,798 parameter vectors evaluated during three GA force field parameter optimization runs. To obtain a homogeneous data set with parameter vectors pertaining to rms values relevant to optimization, only those parameter vectors have been used that had rms values within the range 0.3–0.5 Å. To perform backpropagation training, these rms values have been normalized between 0 and 1: $\mathrm{rms}_{norm} = [(\mathrm{rms}_{orig} - \mathrm{rms}_{min})/(\mathrm{rms}_{max} - \mathrm{rms}_{min})]$ with $\mathrm{rms}_{max} = 0.5$ and $\mathrm{rms}_{min} = 0.3$.

GA REFINEMENT

The program GAPAO, based on the PGA Program Package,[19] and described elsewhere,[8] was used to optimize force field parameters by GA. In the optimization described earlier, an elitist optimization strategy[8] and uniform crossover operator[20] were used. Fitness scaling was chosen to be linear.

FORCE FIELD CALCULATION

A modified version of the program YAMMP[21] was used throughout. The original version of YAMMP was modified[8] so as to reproduce exactly the results of the mm2* force field as implemented in the commercial program package MACROMODEL.[14] This modified program was embedded in shells that allow for its use in different algorithmic environments and also on parallel computing equipment.[8] For purposes of analysis of the behavior of the GA optimization procedure with the number and type of parameters, an augmented set of variable parameters with respect to the one described as mm2*f* in ref. 8 was used. The variable parameters were refined as shown in Table III.

The values obtained after optimization based on the force field evaluator are labeled mm2*a*. The values obtained by applying the trained neural network (see earlier) as the evaluator are labeled mm2*n*. It is seen that some of the values are

TABLE III.
Definition of Force Field Parameters Allowed to Vary.

(a) Bond stretching: $E_B = \frac{1}{2} \cdot k_b \cdot [(r - r_0)^2 - 2 \cdot (r - r_0)^3]$

Force field parameter	mm2*n*		mm2*a*	
	r_0	k_b	r_0	k_b
Mo — C1	1.93*	2.00	1.97*	2.00
Mo — P	2.44*	0.41*	2.57*	0.41*

(b) Angle bending: $E_A = \frac{1}{2} \cdot k_a \cdot [(\alpha - \alpha_0)^2 + \mathrm{const} \cdot (\alpha - \alpha_0)^6]$

Force field parameter	mm2*n*		mm2*a*	
	α_0	k_a	α_0	k_a
P — Mo — P	84.3*	1.01*	83.0*	1.24*
C1 — Mo — C1	83.0*	2.00*	88.3*	0.14*
C1 — Mo — P	94.9 / 175.9	0.27*	94.9 / 175.9	1.2*
Mo — C1 — O	170.0*	0.08*	170.0*	0.06*
Mo — P — C2	123.0	0.27*	119.7*	0.28*
Mo — P — C3	113.7*	0.01*	115.7*	0.01*

The values in columns in mm2*a* and mm2*n* represent the estimates after convergence of the refinement process. The mm2*a* set results from GA optimization based on the force field evaluator. The mm2*n* set refers to the same type of optimization with the trained neural network used as evaluator. Refined parameters are indicated with an asterisk (r_0 [Å], k_b [mdyn · Å^{-1}], α_0 [°], k_a [mdyn · Å · rad^{-2}]).

closely similar for both runs, whereas others are substantially different. Because the overall quality of fit is almost the same for both sets (Table II) the multiple minima of almost equal quality must exist on the hypersurface, rms = $f(p)$. With respect to the quality of determining optimal conformations of tripod metal compounds this is not a serious drawback.[8]

COMPUTING ENVIRONMENT

GA optimizations were run in parallel computing environments. A Parsytec GC/Power Plus-192 (192 Power PC 601, 80 MHz, 32 MB/processor) or a Cray T3E-512 with 512 computation nodes (600 MFlops/processor, 128 MB/processor, 333 MHz) were used. The time required for the Cray supercomputer to optimize 14 parameters based on the data set of 10 compounds (Table I) using 64 processors in parallel is 20 hours if the force field evaluator is used. The time required on the same equipment, but with only a single processor, for the same problem (convergence after 200 generations) is only 2 minutes if the fitness criterion is evaluated by the trained neural network.

Neural network training was performed on a Silicon Graphics Indigo^2 MIPS R4400 workstation (200 MHz, 128 MB RAM), using Irix 5.3.

The generation of the 50×50 Kohonen map with the training parameters given, based on the 19,798 14-component vectors, takes about 30 minutes. Training the backpropagation network described herein with 19,798 training vectors to reproduce rms = $f(p)$ takes 5 hours. All of the time values quoted refer to the total time elapsed until the job was complete and not simply CPU time (i.e., time includes all handling necessary in each specific environment to produce the results). The entire number of processors quoted in each case was allocated to the computation over the whole time.

Acknowledgments

The authors are grateful to J. Gasteiger for allowing us to use the program, KMAP, written by the Erlangen group. Financial support by the Deutsche Forschungsgemeinschaft and by the Höchsleistungsrechenzentrum Jülich (computing time on the CRAY T3E parallel computer) is gratefully acknowledged. The Interdisziplinäres Zentrum für Wissenschaftliches Rechnen Heidelberg was of great help in allowing us to do some test computations on its PARSYTEC parallel computer. One of us (J.H.) is indebted to the Graduiertenkolleg Modellierung und Wissenschaftliches Rechnen in Mathematik und Naturwissenschaften for membership.

References

1. See, for example: (a) Collman, J. P.; Hegedus, L. S.; Norton, J. R.; Finke, R. G. Principles and Applications of Organotransition Metal Chemistry; University Science Books: Hill Valley 1987. (b) Brintzinger, H.; Fischer, D.; Muhlhaupt, R.; Rieger, B.; Waymouth, R. M. Angew Chem Int Ed Engl 1995, 34, 1143. (c) Landis, C. R.; Halpern, C. R. J Am Chem Soc 1987, 109, 1746. (d) Helmchen, G.; Kudis, S.; Sennhenn, P.; Steinhagen, H. Pure Appl Chem 1997, 69, 513.
2. Allinger, N. L. J Am Chem Soc 1977, 99, 8127.
3. Burkert, U.; Allinger, N. L. Molecular Mechanics, ACS Monograph; American Chemical Society: Washington, DC, 1982.
4. Comba, P.; Hambley, T. W. Molecular Modelling of Inorganic Compounds; VCH: Weinheim, 1995.
5. Hay, B. J. Coord Chem Rev 1993, 126, 177.
6. Landis, C. R.; Root, D. M.; Cleveland, T. In Reviews in Computational Chemistry; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH: New York, 1995; Vol 6, p 73.
7. See, for example, Bowen, P.; Allinger, N. In Reviews in Computational Chemistry; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH: New York, 1991; Vol 6, p 81.
8. Hunger, J.; Beyreuther, S.; Huttner, G.; Allinger, K.; Zsolnai, L. Eur J Inorg Chem 1998, 6, 693.
9. Beyreuther, S.; Hunger, J.; Huttner, G.; Mann, S.; Zsolnai, L. Chem Ber 1996, 129, 745.
10. Beyreuther, S.; Hunger, J.; Cunskis, S.; Diercks, T.; Planker, E.; Frick, A.; Huttner, G. Eur J Inorg Chem, in press.
11. Beyreuther, S.; Frick, A.; Hunger, J.; Huttner, G. Eur J Inorg Chem, in press.
12. Holland, J. Adaption in Natural and Artificial Systems; MIT Press: Cambridge, MA, 1975.
13. Goldberg, D. E. Genetic Algorithms in Search, Research and Machine Learning; Addison-Wesley: New York, 1989.
14. Mohamadi, F.; Richards, N. G. J.; Guida, W. C.; Liskamp, R.; Lipton, M.; Caufield, C.; Chang, G.; Hendrickson, T.; Still, W. C. J Comput Chem 1990, 11, 440.
15. Zupan, J.; Gasteiger, J. Neural Networks in Chemistry; VCH: Weinheim, 1993.
16. Zell, A. Simulation Neuronaler Netze; Addison-Wesley: Bonn, 1994.
17. Kohonen, T. Self-Organization and Associatie Memory, 2nd ed.; Springer: Berlin, 1988.
18. Zell, A., et al. SNNS User Manual, Version 4.1, 1995. The program is available at http://vaserely.stuttgart.de.
19. Levine, D. Argonne National Laboratory, 1995. The program is available at http://www.mcs.anl.gov/pgapack.html.
20. Syswerda, G. Proceedings of the 3rd International Conference on Genetic Algorithms; Morgan Kaufman: San Mateo, 1989.
21. Tan, R. K. Z.; Harvey, S. C. J Comput Chem 1993, 14, 455.